Welcome

Solving Quadratic Models with IBM ILOG CPLEX Optimization Studio

John Gregory
Senior Product Manager
IBM ILOG CPLEX Optimizer
IBM Software Group

Presenter:

John Gregory
Senior Product Manager
IBM ILOG CPLEX Optimizer
jgregor@us.ibm.com

Moderator:

Gabriela Wilson
Optimization Campaign Specialist

Welcome

Solving Quadratic Models with IBM ILOG CPLEX Optimization Studio

John Gregory
Senior Product Manager
IBM ILOG CPLEX Optimizer
IBM Software Group

Outline

- The quadratic problems solved by IBM ILOG CPLEX
- Fundamental differences between linear and quadratic models
- Algorithms for solving quadratic models
- Modeling interfaces in CPLEX for quadratic problems
- Use of quadratic models in industry
- Some tuning tips

Quadratic problems – QP and MIQP

A mixed integer quadratic program (MIQP) **is an optimization problem of the form**

$$\begin{aligned} &\mathit{Minimize} && c^T x + x^T Q x \\ &\mathit{Subject\ to} && Ax = b \\ & && l \leq x \leq u \end{aligned}$$

$$\text{some or all } x_j \text{ integer}$$

If no variables are declared to be integer, then the model is simply a quadratic program (QP).

Quadratic problems – QCP and MIQCP

A mixed integer quadratically constrained program (MIQCP) **is an optimization problem of the form**

$$
\begin{aligned}
&\textit{Minimize} && c^T x + \boxed{x^T Q x} \\
&\textit{Subject to} && Ax = b \\
& && a_i^T x + x^T Q_i x \le r_i \qquad i = 1, \ldots, q \\
& && l \le x \le u \\
&\text{some or all } x_j \text{ integer}
\end{aligned}
$$

With no integer variables, then it is simply a QCP.

Quadratic problems - SOCP

- **SOCP: Second Order Cone Programs**
- The above QCP formulation does not make it obvious that SOCP is supported.
- However, CPLEX actually solves QCP problems and MIQCP subproblems by converting internally to an SOCP formulation.
- You can solve SOCP, if of the form
  - $x'Qx <= y^2$ where $y >= 0$ and Q is PSD, or
  - $x'Qx <= yz$ where $y >= 0$, $z >= 0$, and Q is PSD
- In other words, **QCP provides SOCP** in a **more general** framework. CPLEX solves SOCP.

Quadratic problems - Convexity

- CPLEX solves MIQP and MIQCP problems. However, all the quadratic functions must be **Convex**.

- A convex model carries a guarantee that any “local” optimum is also a “global” optimum.

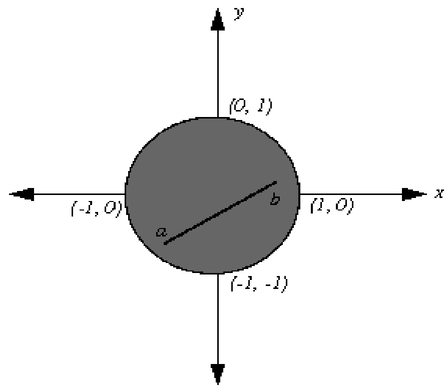
- A convex quadratic matrix has the property of Positive Semi-Definiteness (PSD).

Help - IBM ILOG CPLEX Optimization Studio

Search: convexity GO Search scope: CPLEX

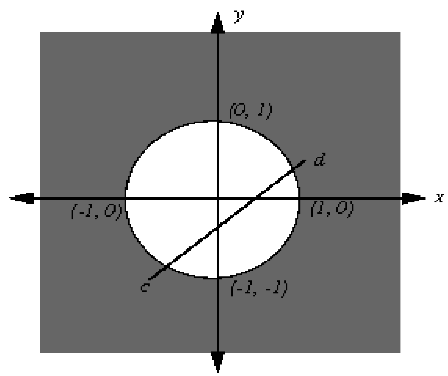
Convexity

The inequality $x^2 + y^2 \leqslant 1$ is convex. To give you an intuitive idea about convexity, the figure titled $x^2 + y^2 \leqslant 1$ is convex graphs that inequality and shades the area that it defines as a constraint. If you consider *a* and *b* as arbitrary values in the domain of the constraint, you see that any continuous line segment between them is contained entirely in the domain.



$x^2 + y^2 \leqslant 1$ is convex

The inequality $x^2 + y^2 \geqslant 1$ is not convex; it is concave. The figure titled $x^2 + y^2 \geqslant 1$ is not convex graphs that inequality and shades the area that it defines as a constraint. If you consider *c* and *d* as arbitrary values in the domain of this constraint, then you see that there may be continuous line segments that join the two values in the domain but pass outside the domain of the constraint to do so.



$x^2 + y^2 \geqslant 1$ is not convex

Quadratic problems - Convexity

- Some quadratic functions are not convex in either direction, for example **x*y** .
- If your quadratic matrix is not convex, and thus not PSD:
  - Function **CPXqpindefcertificate** (or qpIndefCertificate in Concert) can compute a vector “x” such that x'Qx < 0.
  - Such a vector demonstrates that the matrix Q violates the assumption of positive semi-definiteness, and can be an aid in debugging a user's program if indefiniteness is an unexpected outcome.

Quadratic problems - Convexity

- CPLEX will reformulate under-the-hood where possible, to allow solutions of models that are nominally non-convex
- Example: binary variables allow a convex equivalent

```
    Maximize
  obj: [ 3 x1 * x2 + 4 x1 * x3 + 5 x2 * x3 ] / 2
Subject To
   c1: x1 + x2 + x3 <= 2
Binaries
  x1  x2  x3
```

The objective function becomes (internally)

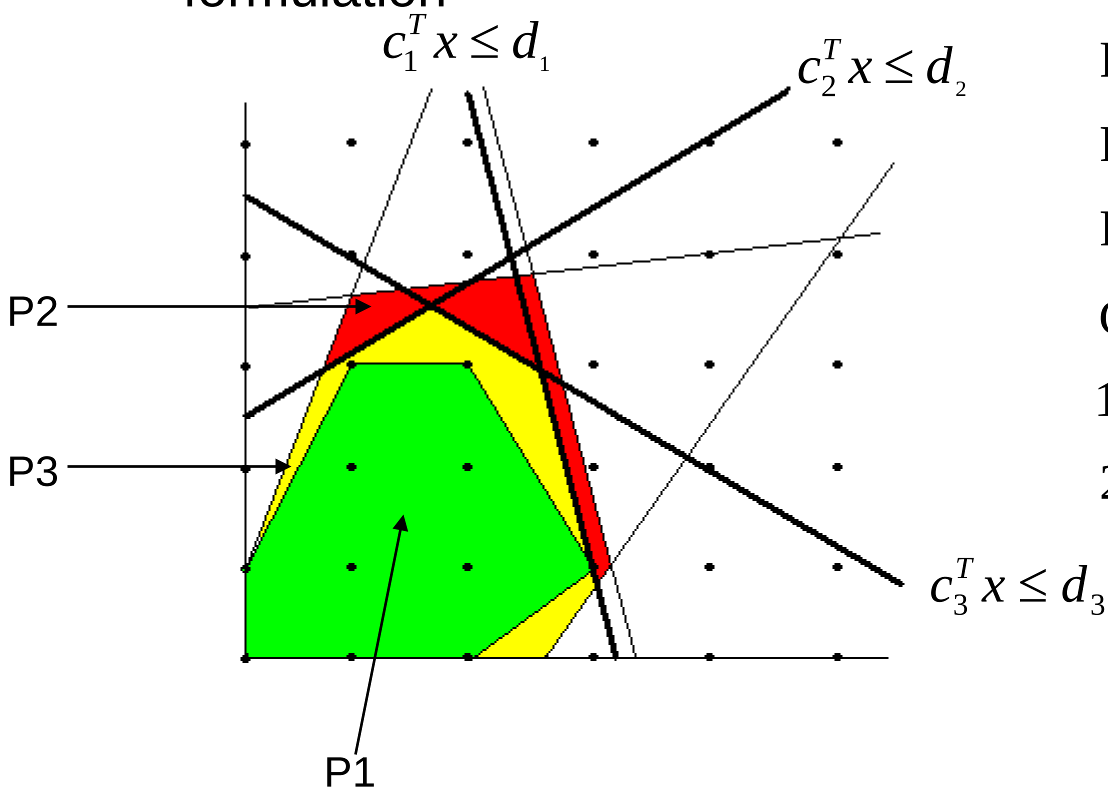
```
    obj: 1.75 x1 + 2. x2 + 2.25 x3
+ [ - 3.5 x1 ^2 + 3 x1 * x2 + 4 x1 * x3 - 4. x2 ^2 + 5 x2 * x3 - 4.5 x3 ^2 ] / 2
```

But this reformulation can not be for a continuous QP!

Differences between MILP and quadratic models

Differences between MILP and quadratic models

- In an MILP Convex hull of the integer feasible solutions provides the strongest formulation
- Add valid cuts based on linear constraints, integrality to strengthen formulation



$$P1 = conv(\{x \in Z^n : Ax \le b, x \ge 0\})$$

$$P2 = \{x \in R^n : Ax \le b, x \ge 0\}$$

$$P3 = P2 \cap \{x \in R^n : Cx \le d\}$$

Cuts must satisfy

1) $c_i^T x \le d_i \quad \forall x \in P1$ (validity)

2) $\exists x \in P2 : c_i^T x > d_i$ (separation)

Differences between MILP and quadratic models

```
max 3x1*x2 + 4x1*x3 + 5x2*x3

subject to

x1 + x2 + x3 <= 2

x1, x2, x3 binary
```

MIQP optimal solution: $\overbrace{\texttt{x2 = x3 = 1}}^{\textbf{Extreme point}}$; obj = 5

QP optimal solution: $\underbrace{\texttt{x1 = x2 = x3 = 2/3;}}_{\textbf{Non-vertex, fractional}}$ obj = 16/3

We cannot tighten this formulation with linear constraints since the integer solutions are extreme points of the relaxation polyhedron

Differences between MILP and quadratic models

- **Numerical stability**
  - Quadratic models are inherently subject to greater numerical instability than their linear counterparts.
  - Recommendation: pay extra attention to your formulation
    - Try to center the solution value of your variables around 1.0
    - Try to center the values of your matrix coefficients around 1.0
    - Try to have your objective function and right-hand side coefficients be of a similar magnitude
    - Where we suggest no more than 6 orders of magnitude spread in data coefficients in a linear model, it may be a good idea to aim for no more than 3 orders of magnitude in a quadratic model
      - 100000 x + y <= 100 might be safe for an LP
      - 100000 x^2 + y^2 <= 100 could be begging for trouble
    - In general, combining small and large quantities in a constraint may be hard to control with precision

Solution Algorithms

Solution Algorithms

- For MIQP and MIQCP models, the CPLEX mixed integer optimizer handles all aspects of the branch & cut tree
  - There are a multitude of user controls, set via parameters, that can tune the performance on challenging MIP models.
  - Useful features apply to all problem classes:
    - Parallel processing for faster solutions
    - Solution pools for storing multiple solutions
    - Conflict refiner for analyzing infeasibilities
    - FeasOpt for automating infeasibility repair
    - Solution polishing for difficult models
    - Modeling constructs such as SOS, Semi-continuous, IfThen and other logic constraints (or indicator constraints)

Solution Algorithms

- For continuous QP models, there is a choice of algorithms, as with LP:
  - Primal Simplex
  - Dual Simplex
  - Barrier (crossover to Simplex is available)
- Barrier is often fastest for QP, but if performance is critical try all three.

- For continuous QCP models, the Barrier solver is the only algorithm choice, and crossover is not possible. Moreover, dual solution information is not available.

Solution Algorithms

- As with linear programs, the Barrier algorithm for both of the quadratic problem classes is parallelized so that there is the potential of faster solution times on computers with multiple cores or CPUs (and shared memory architecture).
- Although neither Simplex algorithm for QP is itself parallelized, you can take advantage of multiple processors by invoking the concurrent solver, which runs both Barrier and Simplex side by side, stopping when either method completes its work.

Solution Algorithms

- A general guidance: QP models often take longer to solve than their LP counterparts, and QCP models often take longer to solve than an associated QP.
- For Mixed Integer problems, this effect can be even more noticeable.
- Performance tuning is also more difficult (or less effective) on quadratic models than for linear counterparts.
- Caution, not undue pessimism, is needed.

Modeling interfaces

Modeling interfaces

- The CPLEX Interactive Optimizer
  - "LP format" is a convenient way to experiment and debug
  - Sample model "qpex.lp" provided with the distribution:

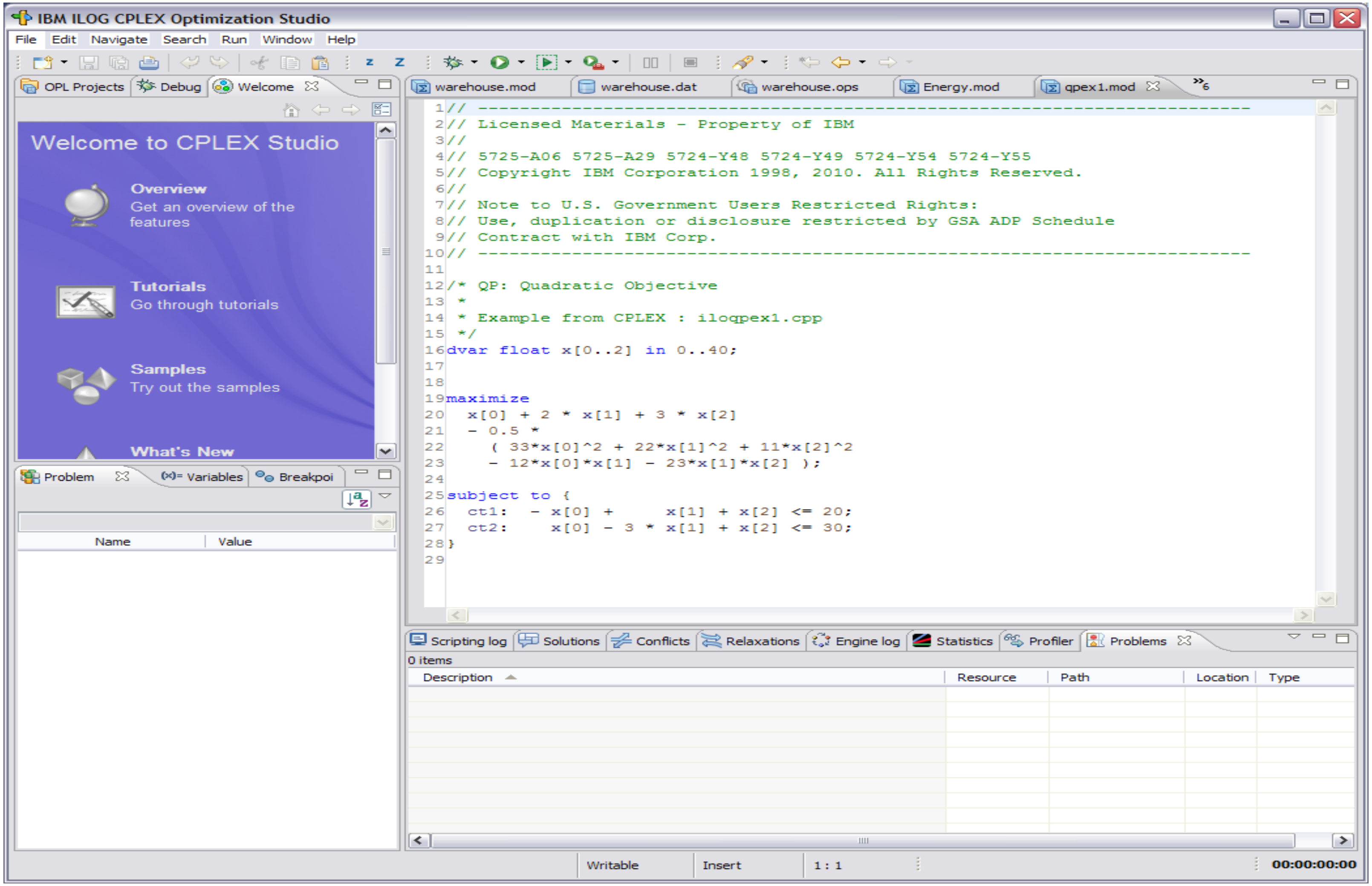
```
Maximize
obj: x1 + 2 x2 + 3 x3 + [ - 33 x1 ^2 + 12 x1 * x2 - 22 x2 ^2 + 23 x2 *
  x3 - 11 x3 ^2 ] / 2
Subject To
 c1: - x1 + x2 + x3 <= 20
 c2: x1 - 3 x2 + x3 <= 30
Bounds
 0 <= x1 <= 40
End
```

Modeling interfaces

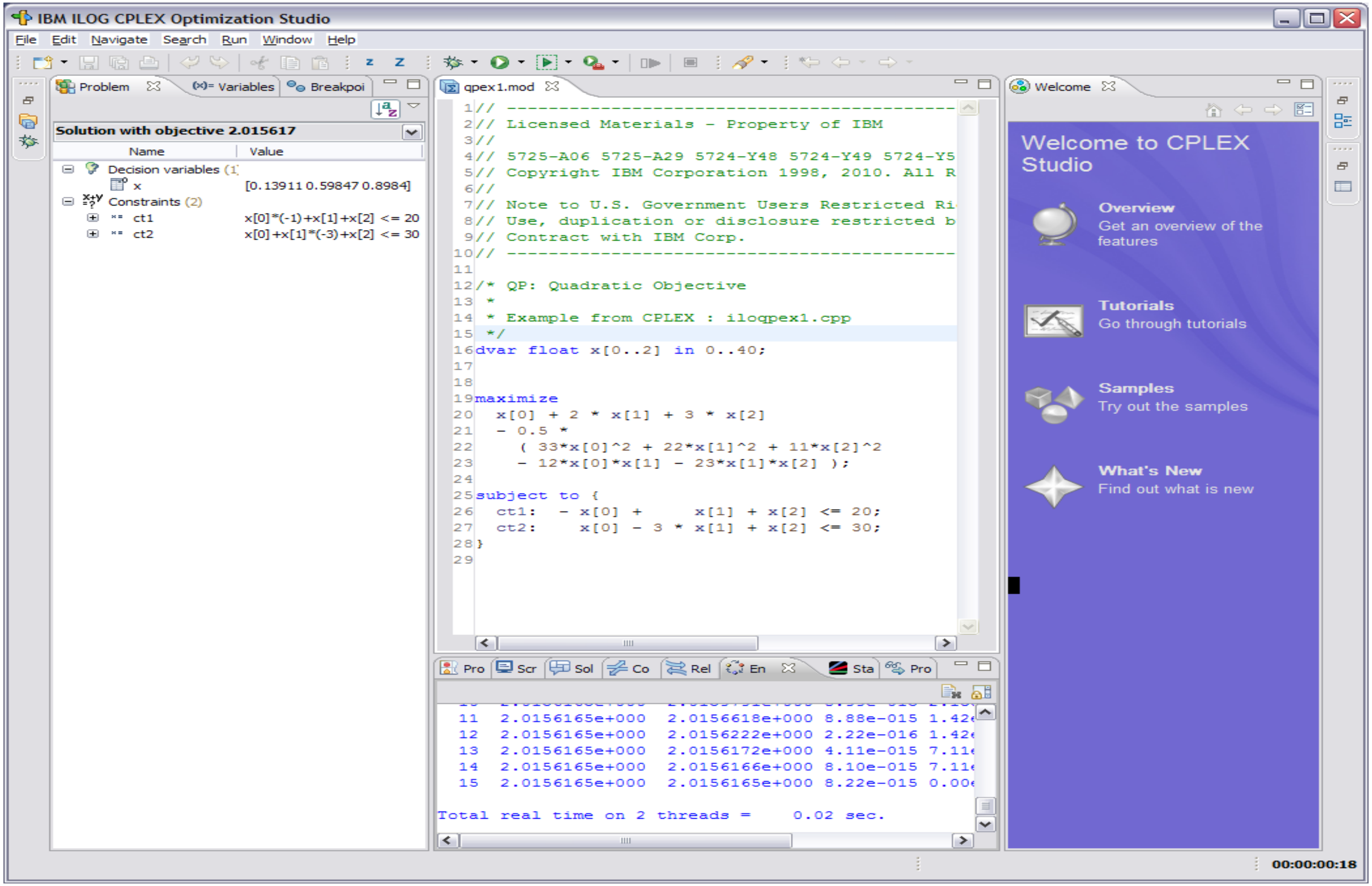
- OPL: a natural language for operations research

```
dvar float x[0..2] in 0..40;
maximize
  x[0] + 2 * x[1] + 3 * x[2]
  - 0.5 *
    ( 33*x[0]^2 + 22*x[1]^2 + 11*x[2]^2
    - 12*x[0]*x[1] - 23*x[1]*x[2] );
subject to {
  ct1:  - x[0] +     x[1] + x[2] <= 20;
  ct2:    x[0] - 3 * x[1] + x[2] <= 30;
}
```

Modeling interfaces – OPL in the CPLEX IDE



Modeling interfaces – OPL in the CPLEX IDE



Modeling interfaces

- C++ Concert provides nearly this same natural style:

```
x.add(IloNumVar(env, 0.0, 40.0));
x.add(IloNumVar(env));
x.add(IloNumVar(env));
model.add(IloMaximize(env, x[0] + 2 * x[1] + 3 * x[2]
                  - 0.5 * (33*x[0]*x[0] + 22*x[1]*x[1] +
                        11*x[2]*x[2] - 12*x[0]*x[1] -
                        23*x[1]*x[2]            ) ));
c.add( - x[0] +     x[1] + x[2] <= 20);
c.add(   x[0] - 3 * x[1] + x[2] <= 30);
model.add(c);
```

Modeling interfaces

- Java® Concert is similar but a little less algebraic looking (and C# .NET® Concert offers similar syntax):

```
double[]   lhs = {-Double.MAX_VALUE, -Double.MAX_VALUE};
double[]   rhs = {20.0, 30.0};
double[][] val = {{-1.0,  1.0,  1.0},
                  { 1.0, -3.0,  1.0}};
int[][]    ind = {{0, 1, 2},
                  {0, 1, 2}};
lp.addRows(lhs, rhs, ind, val);
IloNumExpr x00 = model.prod( 33.0, x[0], x[0]);
IloNumExpr x11 = model.prod( 22.0, x[1], x[1]);
IloNumExpr x22 = model.prod( 11.0, x[2], x[2]);
IloNumExpr x01 = model.prod(-12.0, x[0], x[1]);
IloNumExpr x12 = model.prod(-23.0, x[1], x[2]);
IloNumExpr Q   = model.prod(0.5, model.sum(x00, x11, x22, x01, x12));
double[] objvals = {1.0, 2.0, 3.0};
model.add(model.maximize(model.diff(model.scalProd(x, objvals), Q)));
```

Modeling interfaces

- The Python interface:

```
def setproblemdata(p):
    p.objective.set_sense(p.objective.sense.maximize)
    p.linear_constraints.add(rhs = [20.0, 30.0], senses = "LL")
    obj  = [1.0, 2.0, 3.0]
    ub   = [40.0, cplex.infinity, cplex.infinity]
    cols = [[[0,1],[-1.0, 1.0]],
            [[0,1],[ 1.0,-3.0]],
            [[0,1],[ 1.0, 1.0]]]
    p.variables.add(obj = obj, ub = ub, columns = cols,
                    names = ["one", "two", "three"])
    qmat = [[[0,1],[-33.0, 6.0]],
            [[0,1,2],[ 6.0,-22.0, 11.5]],
            [[1,2],[ 11.5, -11.0]]]
    p.objective.set_quadratic(qmat)
```

Modeling interfaces

- The Callable Library: direct access to CPLEX sparse structures from any language

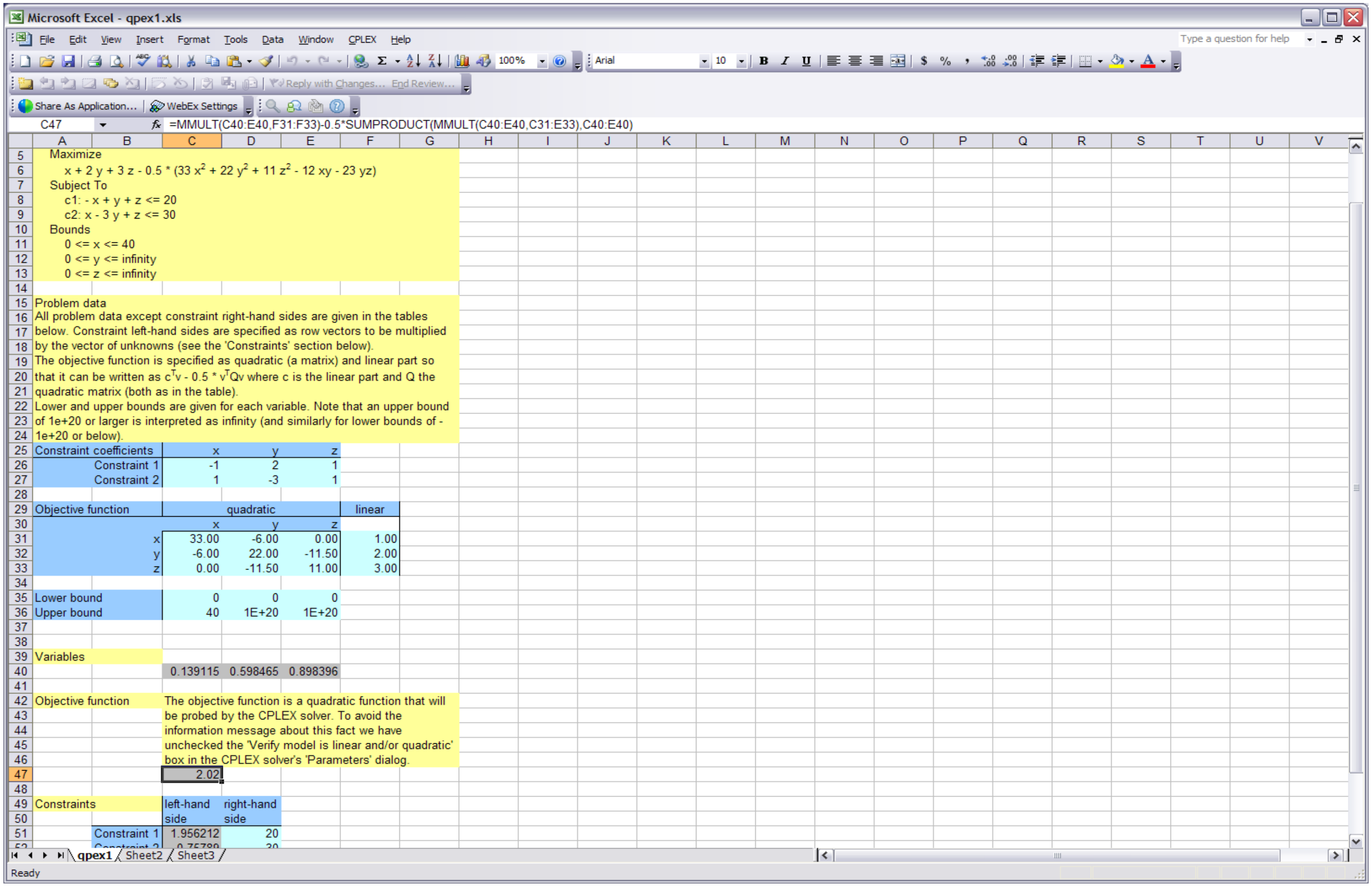
```
zobj[0]  = 1.0;   zobj[1]   = 2.0;   zobj[2] = 3.0;
zmatbeg[0] = 0;     zmatbeg[1] = 2;    zmatbeg[2] = 4;
zmatcnt[0] = 2;     zmatcnt[1] = 2;    zmatcnt[2] = 2;
zmatind[0] = 0;     zmatind[2] = 0;    zmatind[4] = 0;     zsense[0] = 'L';
zmatval[0] = -1.0;  zmatval[2] = 1.0;  zmatval[4] = 1.0;   zrhs[0]   = 20.0;
zmatind[1] = 1;     zmatind[3] = 1;    zmatind[5] = 1;     zsense[1] = 'L';
zmatval[1] = 1.0;   zmatval[3] = -3.0; zmatval[5] = 1.0;   zrhs[1]   = 30.0;
   zlb[0] = 0.0;       zlb[1] = 0.0;          zlb[2] = 0.0;
   zub[0] = 40.0;      zub[1] = CPX_INFBOUND; zub[2] = CPX_INFBOUND;
zqmatbeg[0] = 0;     zqmatbeg[1] = 2;     zqmatbeg[2] = 5;
zqmatcnt[0] = 2;     zqmatcnt[1] = 3;     zqmatcnt[2] = 2;
zqmatind[0] = 0;     zqmatind[2] = 0; zqmatval[0] = -33.0; zqmatval[2] = 6.0;
zqmatind[1] = 1;     zqmatind[3] = 1;     zqmatind[5] = 1;
zqmatval[1] = 6.0;   zqmatval[3] = -22.0; zqmatval[5] = 11.5;
zqmatind[4] = 2;     zqmatind[6] = 2; zqmatval[4] = 11.5;  zqmatval[6] = -11.0;
```

Modeling interfaces

- CPLEX for MATLAB® provides a convenient matrix-oriented toolbox for MATLAB users, combined with the most powerful solution algorithms for MIQP and MIQCP

```
function populatebyrow()
  cplex.addCols([1 2 3]', [], [0 0 0]', [40 Inf Inf]');
  cplex.Model.Q   = [-33     6   0; ...
    6   -22  11.5; ...
    0  11.5 -11];
  cplex.addRows(-inf, [-1  1 1], 20);
  cplex.addRows(-inf, [ 1 -3 1], 30);
end
```

Modeling interfaces - CPLEX for Microsoft Excel®



Modeling interfaces

- In summary, programming (and other) interfaces for:
  - OPL and the CPLEX IDE
  - C++
  - C#
  - Java
  - Excel
  - MATLAB
  - Microsoft Solver Foundation
  - Python
  - Visual Basic
  - Callable Library, for C and other languages
  - Interactive Optimizer

Uses of Quadratic models

Uses of Quadratic models

- Portfolio models in finance
  - Classic Markowitz portfolio models **minimize risk**, formulated as the standard deviation on return, subject to constraints on the least return that is acceptable. This is a QP formulation, and if lot sizes or cardinality constraints are included, then it is a MIQP problem.
  - A natural and sometimes preferable formulation, with the advent of an industrial strength QCP algorithm, reverses the logic and seeks to **maximize the return on investment**, subject to constraints on the amount of risk that is acceptable. The quadratic expression of risk is thus found in the constraints. Again, if discrete variables are added, then the problem becomes a MIQCP.

Uses of Quadratic models

- Portfolio models in finance
  - When these models turn out to be difficult, it is sometimes due either to a great deal of symmetry in the model or to small differences among the data points that must be explored in nearly brute-force fashion.
  - If difficulty occurs during an early and simply formulated prototype, it may seem paradoxical, but the inclusion of additional (realistic) complexity in the model may help break these symmetries or near-ties in decisions, and make the model easier.
  - Of course, the additional size could make the model harder too.
  - But don’t give up automatically, if your simplest prototype looks daunting. “Clean” quadratic models can be among the most difficult.

Uses of Quadratic models

- Electric power industry
  - Direct current (DC) optimal power flow problems. Alternating current systems give rise to nonlinearities not modeled by quadratics, but the DC problem, which seeks to give low costs per kilowatt generated, avoids this complexity. Bid-offer systems have been modeled using convex quadratic formulations. If the demands are fixed, then convexity is assured.
  - Power system reliability models involve a similar risk function to the portfolio model and can use QP or QCP solution techniques.
  - Site selection for electric substations can be models with convex quadratics.

Uses of Quadratic models

- Optimization under uncertainty
  - Again related to the risk measure seen in portfolio models, any system operating under a degree of uncertainty could have its solution robustness measured by a quadratic expression of the standard deviation

Some Performance Tuning Tips

Additional methods to find solutions

- Some MILP tuning tactics are effective for MIQPs as well, even when MIQP feasible region is a convex hull of integer feasible solutions
  – Set MIP emphasis parameter to 1 or 4
  – Aggressive probing may still deduce variable fixings
  – Aggressive strong branching
  – Use feasopt to find starting solution
  – Aggressive use of RINS heuristic, solution polishing to improve upon existing feasible solutions
  – Linear cuts less likely to be effective
    - Feasible region may be convex hull of integer feasible solutions
    - QP relaxation solutions need not be extreme points
- Determine if associated MILP is also difficult to solve
  – If MIQP feasible region is convex hull, MILP will be easy to solve
  – If not, MILP performance tuning tactics will apply to MIQP

Additional methods to find solutions

- Adding a quadratic term to the objective doesn't change the feasible region
  - Solving associated MILP can provide a feasible solution for the MIQP
    - Just remove Q from the objective
    - If Q is diagonal use the diagonals in a linear objective
    - CPLEX's solution can generate multiple MIP starts for the MIQP

$$e = (1,1,\ldots,1),\ Q \text{ diagonal}$$

$$Let\ q = Qe \qquad \text{//extract diagonal elements}$$

$$Minimize \quad c^T x + \boxed{q^T x}$$

$$Subject\ to \quad Ax = b$$

$$l \le x \le u$$

$$\text{some or all } x_j \text{ integer}$$

Additional methods to find solutions

- Adding a quadratic term to the objective doesn't change the feasible region
  - Solving associated MILP can provide a feasible solution for the MIQP
    - If Q has off diagonal elements, we may be able to solve an associated MIQP with a diagonal quadratic matrix.

$$Q = L^T L$$

$$Let\ y = Lx$$

$$\begin{aligned}
&Minimize && c^T x + e^T (y^+ + y^-) \\
&Subject\ to && Ax = b \\
& && (y^+ - y^-) - Lx = 0 \\
& && l \le x \le u \\
& && y^+, y^- \ge 0
\end{aligned}$$

$$\text{some or all } x_j \text{ integer}$$

Dealing with a dense quadratic term

- A fully dense Q matrix can be bad news for solution speed.
- Sometimes, such a Q is derived from sparse data source.
- A common such example: Q is the product of a sparse or triangular matrix times itself, as in the previous example
  - Then let y=Lx, and the quadratic term in the objective function becomes just a diagonal on y.
  - Any remaining density of L is now found among the constraints, where the linear algebra is better equipped to handle it.
- Usually (though not always) density is an indication that the most basic data isn't being used. It's a lot of work to manage millions of independent pieces of input data, and maybe that work isn't really going on, and maybe you can reformulate to use only that basic data.

Linearizing

- When quadratic objective terms all involve at least one binary variable, we can linearize the MIQP into a MILP

```
max 3x1*x2 + 4x1*x3 + 5x2*x3
subject to
x1 + x2 + x3 <= 2
x1, x2, x3 binary
```

```
Let x1*x2 = z12
z12 <= x1
z12 <= x2
z12 + 1 >= x1 + x2
z12 binary
```

Linearizing

- Linearized MIQP:

```
max 3z12 + 4z13 + 5z23        subject to
x1 + x2 + x3 <= 2
z12 <= x1
z12 <= x2
z12 + 1 >= x1 + x2
z13 <= x1
z13 <= x3
z13 + 1 >= x1 + x3
z23 <= x2
z23 <= x3
z23 + 1 >= x2 + x3
All variables binary
```

Linearizing

- Linearizing in this manner is a relaxation of the MIQP
- But, solving the associated MILP brings into play MIP tightening features (especially cuts) that are more likely to be effective

Troubleshooting

- Indefinite Q matrices
  - CPLEX will convexify the objective if all quadratic terms have only binary variables
  - Otherwise, identifying the cause of indefiniteness can be challenging
    - CPXqpindefcertificate/IloCplex::qpIndefCertificate
    - Sample program on CPLEX FAQ site to find minimal indefinite submatrix
    - Use MATLAB to extract eigenvalues of Q

Troubleshooting

- Indefinite Q matrices
  - Use MATLAB to extract eigenvalues of Q
    - At least one eigenvalue of an indefinite Q matrix will be negative
    - Examine minimum negative eigenvalue
      - Determine if Q is indefinite due to round off error, or truly indefinite under perfect precision

```
function qpeig(varargin)
cpx = Cplex();
cpx.readModel(varargin{1});
lambda = eig(cpx.Model.Q);
disp('The minimum eigenvalue of Q matrix is:')
disp(min(lambda))
```

Summary and Conclusions

- MIQPs have fundamental differences from MILPs that can make them potentially more challenging to solve to optimality
- MIQP and MILP have same feasible region
  - MILP approximations to find good starting solutions for MIQP
- MIQPs with binaries in all quadratic terms can be linearized
  - Relaxes the model, but additional MILP features of branch and cut can compensate
- Cuts for MIQP and MIQCP are a growing research area
- CPLEX's MATLAB API extends the debugging and troubleshooting functionality of the interactive CPLEX optimizer, both for MIQPs and in general

In closing…

- For much more detail, consult the CPLEX Optimization Studio documentation, particularly the User's Manual.
- We are interested to know your modeling needs concerning QP convexity, and for nonlinear problems generally. Contact me, John Gregory, at jgregor@us.ibm.com.
- Thank you very much for attending.

POLLING QUESTIONS

QUESTIONS & ANSWERS

धन्यवाद
Hindi

多謝
Traditional Chinese

Grazie
Italian

ขอบคุณ
Thai

Gracias
Spanish

Thank You

多谢
Simplified Chinese

Спасибо
Russian

Obrigado
Brazilian Portuguese

شكراً
Arabic

Danke
German

Merci
French

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다